Федеральное государственное автономное  
образовательное учреждение  
высшего образования  
“СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ”

Институт инженерной физики и радиоэлектроники

Базовая кафедра радиоэлектронной техники информационных систем

**Информатика  
ОТЧЕТ ПО КУРСОВОЙ РАБОТЕ**  
Вариант 1

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_ Боев Н. М.

Подпись, дата

Студент РФ22-21Б 052214690 \_\_\_\_\_\_\_\_\_\_\_\_ Соболев Н. С.

Подпись, дата

Красноярск 2022

**1 Задание**

Написать программу на языке Си, реализующую следующие функции:

1. Чтение бинарных данных из файла (входной сигнал, тип float).
2. Запись бинарных данных в файл (выходной сигнал, тип float).
3. Динамическое выделение памяти для сигналов (функция malloc).
4. Чтение коэффициентов КИХ-фильтра из текстового файла (int32\_t).
5. Фильтрация входного сигнала КИХ-фильтром.
6. Вывод графиков входного и выходного сигналов на экран с использованием псевдографических символов.
7. Генерация бинарного файла, содержащего выборки входного сигнала.
8. Генерация текстового файла с коэффициентами КИХ-фильтра.

В программе должен быть реализован разбор аргументов командной строки (argc, \*argv). Пример запуска программы:  
./fir -i input.bin -o output.bin -f ir.txt  
где input.bin – файл с входным сигналом;  
output.bin – файл с выходным сигналом;  
ir.txt – файл с импульсной характеристикой фильтра.

Для проверки работы программы реализовать возможность генерации полигармонического сигнала, содержащего пять гармонических составляющих A\*sin(w \* n \* t + f), n = 1, 2, 3, 4, 5. Фаза и амплитуда каждой составляющей – случайные (использовать генератор случайных чисел).  
Сигнал должен содержать два периода гармонического колебания с наименьшей частотой. Пример запуска программы для генерации входного сигнала заданного размера (300 выборок сигнала):  
./fir -g input.bin -s 300

Реализовать функцию генерации текстового файла с коэффициентами КИХ-фильтра (все коэффициенты принять равными 1/N, длину N задавать параметром):  
./fir -c ir.txt -n 20

**2 Блок-схема**

Для всех блок-схем были созданы новые типы данных:  
FILE\_SIZE\_TYPE = int  
SIGNAL\_TYPE = float

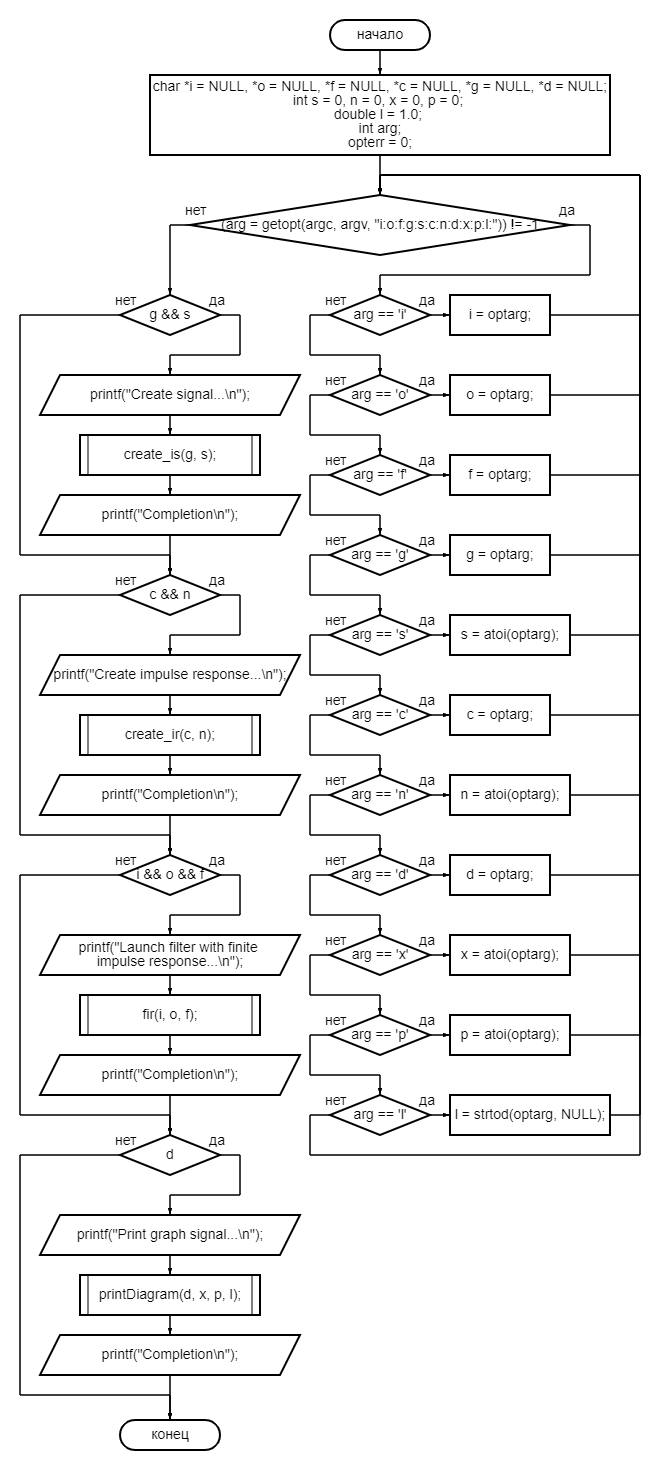


Рисунок 1 – Блок-схема главного алгоритма программы

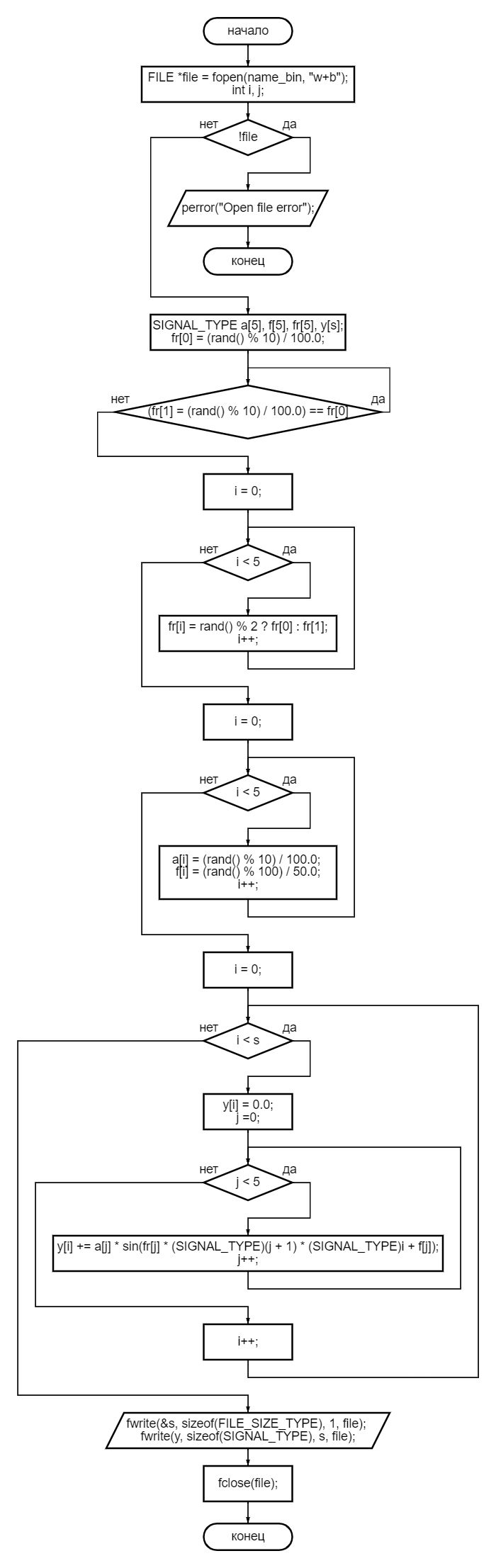


Рисунок 2 – Блок-схема алгоритма генерации полигармонического сигнала

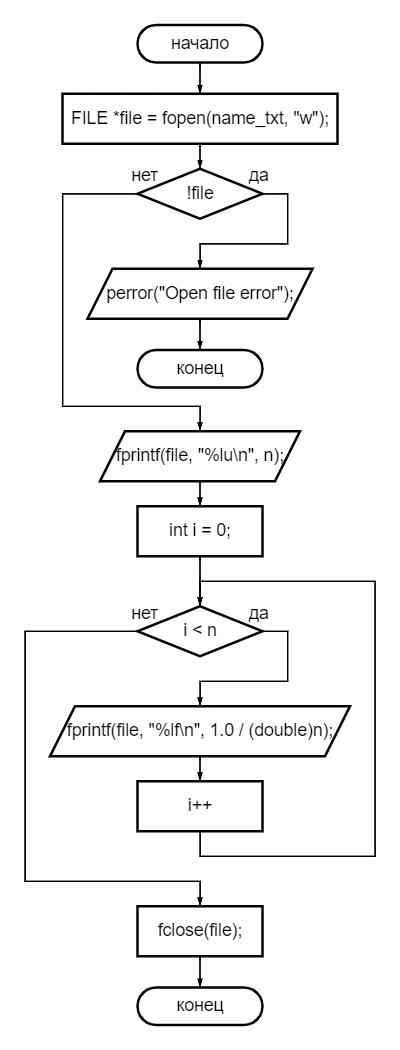


Рисунок 3 – Блок-схема алгоритма генерации коэффициентов КИХ-фильтра

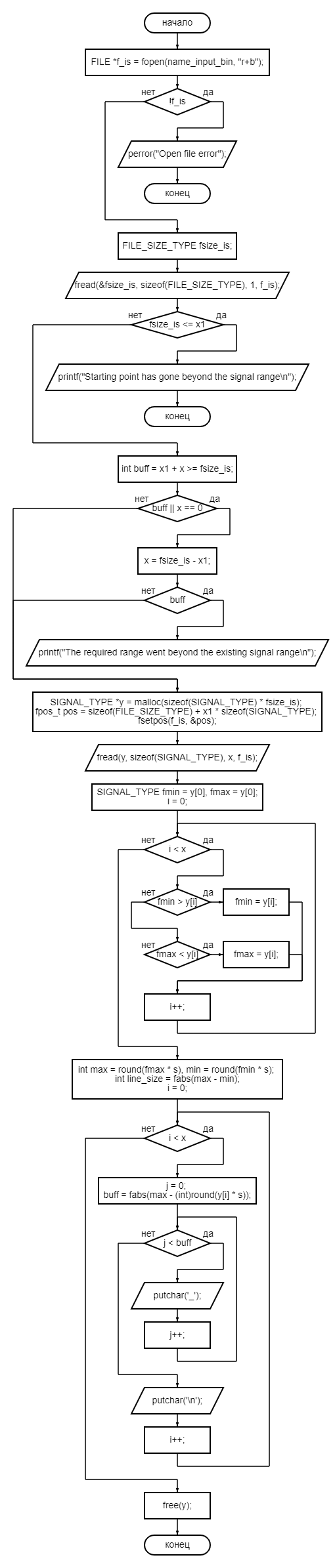


Рисунок 4 – Блок-схема алгоритма вывода графика сигнала

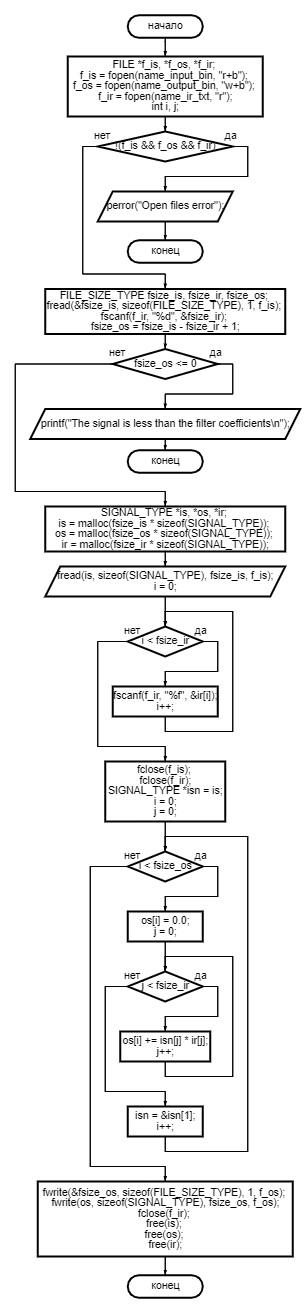


Рисунок 5 – Блок-схема алгоритма КИХ-фильтра

**3 Текст программы**

#include <stdio.h>

#include <math.h>

#include <unistd.h>

#include <stdint.h>

#include <string.h>

#include <stdlib.h>

#define FILE\_SIZE\_TYPE int

#define SIGNAL\_TYPE float

Листинг 1 – Директивы

void main(int argc, char \*argv[])

{

    //Инициализация переменных с нейтральным значением

    char \*i = NULL, \*o = NULL, \*f = NULL, \*c = NULL, \*g = NULL, \*d = NULL;

    int s = 0, n = 0, x = 0, p = 0;

    double l = 1.0;

    int arg;

    opterr = 0; //Запрет вывода optarg ошибок в консоль

    //Поиск аргументов и запись в переменные

    while ((arg = getopt(argc, argv, "i:o:f:g:s:c:n:d:x:p:l:")) != -1)

    {

        switch (arg)

        {

        case 'i':

            i = optarg;

            break;

        case 'o':

            o = optarg;

            break;

        case 'f':

            f = optarg;

            break;

        case 'g':

            g = optarg;

            break;

        case 's':

            s = atoi(optarg); // int

            break;

        case 'c':

            c = optarg;

            break;

        case 'n':

            n = atoi(optarg); // int

            break;

        case 'd':

            d = optarg;

            break;

        case 'x':

            x = atoi(optarg); // int

            break;

        case 'p':

            p = atoi(optarg); // int

            break;

        case 'l':

            l = strtod(optarg, NULL); // double

            break;

        default:

            break;

        }

    }

    //Запуск создания сигнала

    if (g && s)

    {

        printf("Create signal...\n");

        create\_is(g, s);

        printf("Completion\n");

    }

    //Запуск создания импульсной характеристики

    if (c && n)

    {

        printf("Create impulse response...\n");

        create\_ir(c, n);

        printf("Completion\n");

    }

    //Запуск КИХ-фильтра

    if (i && o && f)

    {

        printf("Launch filter with finite impulse response...\n");

        fir(i, o, f);

        printf("Completion\n");

    }

    //Отрисовка графика сигнала

    if (d)

    {

        printf("Print graph signal...\n");

        printDiagram(d, x, p, l);

        printf("Completion\n");

    }

}

Листинг 2 – Код главного алгоритма программы

void create\_is(const char \*name\_bin, const int s)

{

    //Инициализация файла

    FILE \*file = fopen(name\_bin, "w+b");

    //Проверка на файловую ошибку и вывод ошибки

    if (!file)

    {

        perror("Open file error");

        return;

    }

    //Инициализация массивов

    SIGNAL\_TYPE a[5], f[5], fr[5], y[s];

    //Заполнение массива частоты

    fr[0] = (rand() % 10) / 100.0;

    while ((fr[1] = (rand() % 10) / 100.0) == fr[0])

    {

    }

    for (int i = 0; i < 5; i++)

    {

        fr[i] = rand() % 2 ? fr[0] : fr[1];

    }

    //Заполнение массивов фазы и амплитуды

    for (int i = 0; i < 5; i++)

    {

        a[i] = (rand() % 10) / 100.0;

        f[i] = (rand() % 100) / 50.0;

    }

    //Создание полигармонического сигнала из 5 гармонических

    for (int i = 0; i < s; i++)

    {

        y[i] = 0.0;

        for (int j = 0; j < 5; j++)

        {

            y[i] += a[j] \* sin(fr[j] \* (SIGNAL\_TYPE)(j + 1) \* (SIGNAL\_TYPE)i + f[j]);

        }

    }

    //Запись сигнала в фаил

    fwrite(&s, sizeof(FILE\_SIZE\_TYPE), 1, file);

    fwrite(y, sizeof(SIGNAL\_TYPE), s, file);

    fclose(file);

}

Листинг 3 – Код алгоритма генерации полигармонического сигнала

void create\_ir(const char \*name\_txt, const int n)

{

    //Инициализация файла

    FILE \*file = fopen(name\_txt, "w");

    //Проверка на файловую ошибку и вывод ошибки

    if (!file)

    {

        perror("Open file error");

        return;

    }

    //Запись импульсных характеристикик в фаил

    fprintf(file, "%lu\n", n);

    for (int i = 0; i < n; i++)

    {

        fprintf(file, "%lf\n", 1.0 / (double)n);

    }

    fclose(file);

}

Листинг 4 – Код алгоритма генерации коэффициентов КИХ-фильтра

void printDiagram(const char \*name\_input\_bin, int x1, int x, double s)

{

    //Инициализация файла

    FILE \*f\_is = fopen(name\_input\_bin, "r+b");

    //Проверка файловых ошибок и вывод ошибки

    if (!f\_is)

    {

        perror("Open file error");

        return;

    }

    //Чтение длины файла

    FILE\_SIZE\_TYPE fsize\_is;

    fread(&fsize\_is, sizeof(FILE\_SIZE\_TYPE), 1, f\_is);

    //Проверка возможной ошибки выхода из диапазона

    if (fsize\_is <= x1)

    {

        printf("Starting point has gone beyond the signal range\n");

        return;

    }

    //Замена области отрисовки

    int buff = x1 + x >= fsize\_is;

    if (buff || x == 0)

    {

        x = fsize\_is - x1;

        if (buff)

        {

            printf("The required range went beyond the existing signal range\n");

        }

    }

    //Инициализация массива и выгрузка в него сигнала

    SIGNAL\_TYPE \*y = malloc(sizeof(SIGNAL\_TYPE) \* fsize\_is);

    fpos\_t pos = sizeof(FILE\_SIZE\_TYPE) + x1 \* sizeof(SIGNAL\_TYPE);

    fsetpos(f\_is, &pos);

    fread(y, sizeof(SIGNAL\_TYPE), x, f\_is);

    //Поиск минимального и максимального значения сигнала

    SIGNAL\_TYPE fmin = y[0], fmax = y[0];

    for (int i = 0; i < x; i++)

    {

        if (fmin > y[i])

        {

            fmin = y[i];

        }

        else if (fmax < y[i])

        {

            fmax = y[i];

        }

    }

    //Вычисление графика в высоту

    int max = round(fmax \* s), min = round(fmin \* s);

    int line\_size = fabs(max - min);

    //Вывод графика в консоль

    for (int i = 0; i < x; i++)

    {

        int j = 0;

        buff = fabs(max - (int)round(y[i] \* s));

        while (j < buff)

        {

            putchar('\_');

            j++;

        }

        putchar('\n');

    }

    free(y);

}

Листинг 5 – Код алгоритма вывода графика сигнала

void fir(const char \*name\_input\_bin, const char \*name\_output\_bin, const char \*name\_ir\_txt)

{

    //Инициализация файлов

    FILE \*f\_is, \*f\_os, \*f\_ir;

    f\_is = fopen(name\_input\_bin, "r+b");

    f\_os = fopen(name\_output\_bin, "w+b");

    f\_ir = fopen(name\_ir\_txt, "r");

    //Проверка на ошибку и вывод ошибки

    if (!(f\_is && f\_os && f\_ir))

    {

        perror("Open files error");

        return;

    }

    //Запись длины всех файлов

    FILE\_SIZE\_TYPE fsize\_is, fsize\_ir, fsize\_os;

    fread(&fsize\_is, sizeof(FILE\_SIZE\_TYPE), 1, f\_is);

    fscanf(f\_ir, "%d", &fsize\_ir);

    fsize\_os = fsize\_is - fsize\_ir + 1;

    //Проверка на ошибку (длина сигнала меньше длины коэффициентов фильтра)

    if (fsize\_os <= 0)

    {

        printf("The signal is less than the filter coefficients\n");

        return;

    }

    //Инициализация массивов для выгрузки данных с файлов и записи

    SIGNAL\_TYPE \*is, \*os, \*ir;

    is = malloc(fsize\_is \* sizeof(SIGNAL\_TYPE));

    os = malloc(fsize\_os \* sizeof(SIGNAL\_TYPE));

    ir = malloc(fsize\_ir \* sizeof(SIGNAL\_TYPE));

    //Выгрузка файлов в массивы

    fread(is, sizeof(SIGNAL\_TYPE), fsize\_is, f\_is);

    for (int i = 0; i < fsize\_ir; i++)

    {

        fscanf(f\_ir, "%f", &ir[i]);

    }

    fclose(f\_is);

    fclose(f\_ir);

    //Алгоритм КИХ-фильтра

    SIGNAL\_TYPE \*isn = is;

    for (int i = 0; i < fsize\_os; i++)

    {

        os[i] = 0.0;

        for (int j = 0; j < fsize\_ir; j++)

        {

            os[i] += isn[j] \* ir[j];

        }

        isn = &isn[1];

    }

    //Запись результата в фаил

    fwrite(&fsize\_os, sizeof(FILE\_SIZE\_TYPE), 1, f\_os);

    fwrite(os, sizeof(SIGNAL\_TYPE), fsize\_os, f\_os);

    //Зачистка ненужного

    fclose(f\_ir);

    free(is);

    free(os);

    free(ir);

}

Листинг 6 – Код алгоритма КИХ-фильтра

**4 Результат программы**

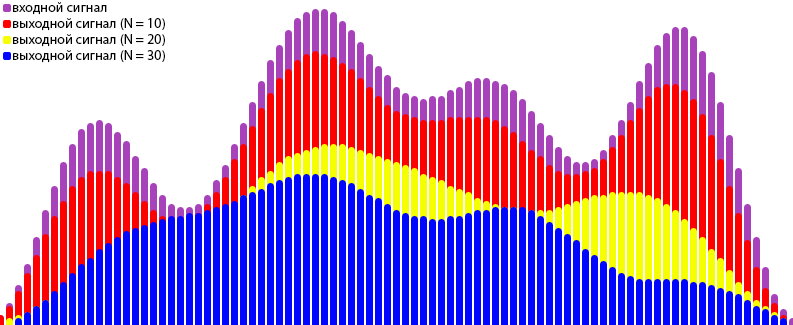


Рисунок 6 – Результат программы

**5 Заключение**

КИХ-фильтры позволяют эффективно избавляться от шумов в сигнале, однако для работы данного фильтра требуется правильно подобранная импульсная характеристика и ресурсы процессора, потребление которых зависит от количества элементов импульсной характеристики.